DevOps Case Study Report

Project: simple-webapp-jenkins-ci-cd | Author: Punith C | Date: November 27, 2025

Executive Summary

This report presents a complete CI/CD pipeline for a static web application using Jenkins. The pipeline automates checkout, validation, testing, and Docker-based deployment. The final build (#10) achieved 100% (4/4 stages) success and deployed a containerized app accessible at http://localhost:8090.

Problem Statement & Objectives

Create a CI/CD pipeline for a static HTML/CSS/JS website using Jenkins and implement automated deployment. Objectives included ensuring reproducible builds, minimal manual steps, and clear pipeline visibility.

Objectives

- Automate Checkout → Build → Test → Deploy
- Validate required files before deployment
- Use Docker/Nginx for reproducible deployment
- Document results and lessons learned

Scope

- Static website: index.html, styles.css, script.js
- Jenkins declarative pipeline (4 stages + post)
- Dockerfile for Nginx-based container
- Optional docker-compose for local Jenkins + webapp

Environment & Architecture

Windows host with WSL2 Ubuntu as the Jenkins agent. Docker used for building and running the containerized web app. Architecture flow: Developer (GitHub) → Jenkins Pipeline → Docker Image → Nginx Container → Browser.

Technologies Used

- HTML5, CSS3, JavaScript
- Nginx (nginx:alpine)
- Docker
- Jenkins (Declarative Pipeline)
- Git/GitHub

Tools Used

- Jenkins 2.528.2
- Docker 28.2.2

- Git 2.43.0
- Windows + WSL2 (Ubuntu 24.04)
- VS Code

Methodology / Implementation Steps

- Create app files and Dockerfile; initialize Git repo
- Configure Jenkins (Pipeline from SCM; set script path)
- Make scripts executable; add Jenkins to docker group; restart Jenkins
- Run validate.sh in Build & Test stages
- Build Docker image and run container on port 8090
- Post-deploy verification and logging

Jenkins Pipeline

- Stages: Checkout, Build (validate), Test (validate), Deploy (docker)
- Environment: DEPLOY_METHOD=docker; CONTAINER_NAME=simple-webapp-demo; WEBAPP_PORT=8090
- Nested repo path handled with dir() blocks

Deployment

- Image: simple-webapp:latest (Image ID: e54b76dbdf5f)
- Container: simple-webapp-demo (ID: 1db05a33b511)
- Port mapping: 8090 → 80; URL: http://localhost:8090
- Automated stop/remove old container before redeploy

Results & Metrics

- Build #10: SUCCESS (4/4 stages)
- Total runtime ~15 seconds
- Application functional; JS alert verified

Screenshots

[Add screenshot: Jenkins Build #10 — All stages green — missing file screenshots/jenkins_build.png]

[Add screenshot: Docker container simple-webapp-demo (8090→80) — missing file screenshots/docker_ps.png]

[Add screenshot: App at http://localhost:8090 with JS alert — missing file screenshots/app_browser.png]

Challenges & Resolutions

- Jenkinsfile path + nested repo → set Script Path + dir()
- Script permissions on Linux → git chmod + pipeline chmod
- Docker permission denied → add Jenkins to docker group + restart
- Image pull timeout → pre-cache nginx:alpine
- Deploy method → switched from sudo copy to Docker

Lessons Learned

- Docker simplifies CI/CD deployments
- Repo structure impacts Jenkins configuration
- Windows→Linux scripts need executable flags
- Group membership changes require service restart

Conclusion

The CI/CD pipeline met all objectives, delivering 100% (4/4 stages) success with a reproducible Docker-based deployment. The project is production-ready and demonstrates a clear, maintainable Jenkins workflow for static sites.

References

- Repo: https://github.com/Punith968/simple-webapp-jenkins-ci-cd
- Jenkins Docs: https://www.jenkins.io/doc/
- Docker Docs: https://docs.docker.com/

Appendix — Key Commands

```
docker build -t simple-webapp:latest .
```

```
docker run -d --name simple-webapp-demo -p 8090:80 simple-webapp:latest
```

```
./validate.sh
```